PixL2R: Guiding Reinforcement Learning Using Natural Language by Mapping Pixels to Rewards

Prasoon Goyal, Scott Niekum, Raymond J. Mooney
Department of Computer Science
University of Texas at Austin
{pgoyal, sniekum, mooney}@cs.utexas.edu

Abstract: Reinforcement learning (RL), particularly in sparse reward settings, often requires prohibitively large numbers of interactions with the environment, thereby limiting its applicability to complex problems. To address this, several prior approaches have used natural language to guide the agent's exploration. However, these approaches typically operate on structured representations of the environment, and/or assume some structure in the natural language commands. In this work, we propose a model that directly maps pixels to rewards, given a free-form natural language description of the task, which can then be used for policy learning. Our experiments on the Meta-World robot manipulation domain show that language-based rewards significantly improves the sample efficiency of policy learning, both in sparse and dense reward settings.

Keywords: reinforcement learning, reward shaping, natural language

1   Introduction

Reinforcement learning (RL) problems often involve a trade-off between the ease of designing a reward function and the ease of learning from this reward. At one end of the spectrum, a sparse reward function – e.g. a fixed positive reward for completing the task, and zero in all other states – is easy to design, but does not give the learning agent any learning signal until it reaches the goal. As such, the agent requires considerable exploration before any learning can take place. At the other end of the spectrum, a dense reward function – e.g. distance to the next waypoint – can be specified to provide the agent with a stronger learning signal, but is often harder to design and tune compared to sparse reward functions. To get around the challenge of reward design, learning from demonstrations is a popular approach [1, 2]; however, providing demonstrations to robots requires teleoperating or kinesthetic teaching, which is difficult and time-consuming to provide, particularly for non-experts. As such, several methods have been proposed recently, which involve guiding an agent using natural language commands which are quick and easy to provide [3].

While promising, these techniques are still quite restrictive, often requiring object properties to be predefined [4, 5], and/or assuming some structure in the natural language commands [6], which is challenging to scale. Other techniques are applicable to only a restrictive set of environments, such as those with discrete action spaces [7]. In this work, we propose a framework that makes no such assumptions, and directly learns to map pixels to rewards for continuous control given a free-form natural language description of the task.

Our approach contains two phases – (1) a supervised learning phase that takes in paired (trajectory, language) data and learns a model of relatedness between a trajectory and a language command, and (2) a policy training phase with a standard RL setup with an additional linguistic description of the task, wherein the relatedness model is used to generate intermediate rewards using the currently executed trajectory and task description.

For instance, consider the domain shown in Figure 1, which is adapted from the recently released Meta-World benchmark [8]. Here, we want the robot to press the green button. Different tasks in this domain require interacting with different objects. In a sparse reward setting, the agent is given a non-zero reward only upon successfully interacting with the pre-selected object. In the absence of
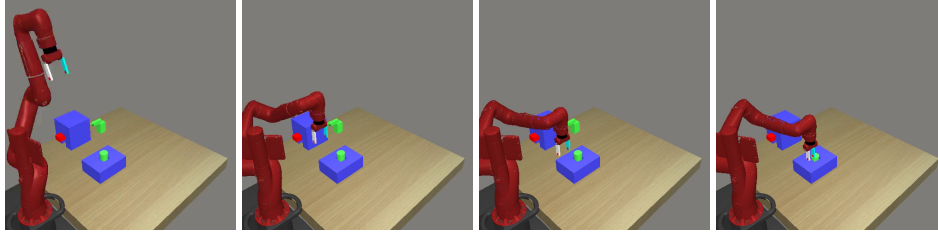
Figure 1: A simulated robot completing a task ("push the green button") in the Meta-World domain.

any other learning signal, the agent will explore randomly until it accidentally completes the desired task. Using natural language to describe the task and generating intermediate rewards from these descriptions can guide the agent towards the goal, significantly speeding up learning.

Our experiments on a diverse set of tasks in the Meta-World domain demonstrate that the proposed approach results in improved sample efficiency during policy learning, both in sparse and hand-designed dense reward settings. This motivates a new paradigm where language could be used to improve over hand-designed rewards, which may be suboptimal owing to the difficulty of designing rewards by hand.

2   Related Work

A number of prior approaches have been proposed to use language to guide a learning agent [9].

Some approaches involve mapping natural language instructions directly to an action sequence to be executed. Tellex et al. [10] dynamically instantiate a graphical model given a language command, from which a plan for the agent is inferred. Sung et al. [11] learn a neural network to predict relatedness between ⟨trajectory, language⟩ pairs and ⟨trajectory, point cloud⟩ pairs, which is then used to find the most likely trajectory given a new language and point cloud. Our approach is different from these approaches in that we use language to generate a reward for the current state, that can then be used to learn a policy using standard RL, which is a more general setting that does not require knowledge of the environment dynamics, and can also work in more complex environments because of the policy learning phase.

Several prior approaches map natural language to a reward function [4, 12, 5], but assume a specific structure of the reward function, while our approach does not make any such assumptions.

Some approaches use a predefined set of linguistic instructions to guide the learning agent. Kuhlmann et al. [13] and Branavan et al. [14] find the most relevant instruction to follow at each state. Kaplan et al. [15] and Waytowich et al. [16] learn a neural network that predicts the similarity between a natural language instruction and a state, and use that to follow a fixed sequence of natural language commands. These approaches use hand-designed features, whereas we propose to learn the association between language and trajectories from a small set of human-provided descriptions.

Some approaches learn to ground language while interacting with the environment [17, 18, 6]. Our approach involves a separate supervised learning phase to ground language, which does not require interacting with the environment.

Fu et al. [19] learn a language-conditioned reward function, but require knowledge of environment dynamics to compute the optimal policy during training. Narasimhan et al. [20] use natural language to transfer dynamics across environments. Blukis et al. [21] generate a state visitation distribution given a natural language instruction, which is then used to generate rewards for policy training. Harrison et al. [22] learn a distribution of states and actions given a natural language command, which is used for policy shaping. Lynch et al. [23] learn a goal-image- and language-conditioned policy, using behavior cloning. Andreas el al. [24] use natural language as a parameter space in supervised and reinforcement learning tasks. Paxton et al. [25] generate subgoals from natural language descriptions of the task, which are then used by a controller to predict actions. Goyal et al. [7] use a similar framework as us, but their approach uses only the actions to generate language-based rewards, without taking into account the states, and requires the action space to be discrete, which is not applicable to most robotics tasks.

Our setting is related to the problem of vision-language navigation (VLN) [26]; while techniques in VLN work with complex multi-step instructions but predominantly discrete action spaces, the approach presented here is applicable to both discrete and continuous action spaces, but currently works with relatively simple instructions. Combining the strengths of both these settings to develop approaches that can work with complex instructions in continuous actions spaces is an interesting direction for future work.

3   Approach

Reinforcement learning consists of an agent interacting with an environment. The learning problem is typically represented using a Markov Decision Process (MDP) $M = \langle S, A, T, R, \gamma \rangle$. Here, S is the set of all states in the environment, A is the set of actions available to the agent, $T : S \times A \times S \to [0, 1]$ is the transition function of the environment, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is a discount factor.

At timestep t, the agent observes a state $s_t \in S$, and takes an action $a_t \in A$, according to some policy $\pi : S \times A \to [0, 1]$. The environment transitions to a new state $s_{t+1} \sim T(s_t, a_t, \cdot)$, and the agent receives a reward $R_t = R(s_t, a_t)$. The goal is to learn a policy π, such that the expected future return, $G_t = \sum_{t=0}^{T} \gamma^t R_t$, is maximized.

In this work, we use an extension of the standard MDP, defined as $M' = \langle S, A, T, R, \gamma, L \rangle$, where L is an instruction describing the task using natural language, and the other quantities are as defined above. We use the following two-phase framework for learning in an MDP with a natural language description of the task (Figure 2).

Phase 1: A neural network (PixL2R) is trained to predict whether a given trajectory and language are related or not. This requires paired ⟨trajectory, language⟩ data in the environment. We describe this phase in detail in Section 3.1.

Phase 2: Next, a policy is trained for a new task – in addition to the extrinsic reward from the environment, the agent additionally gets a language command describing the task. At every step, the agent's trajectory so far is compared against the description of the task using the trained PixL2R model and the relatedness scores predicted by the model are used to generate intermediate rewards for reward shaping [27]. Section 3.2 describes this phase.

Figure 2: Overview of the approach

Note that the trained PixL2R model can be used during policy learning for a wide variety of downstream tasks, insofar as the objects and linguistic vocabulary in these tasks closely match the data used to train the PixL2R model. Thus, the cost of training PixL2R is amortized across all the downstream tasks.

3.1   PixL2R: Pixels and Language to Reward

First, a relatedness model – PixL2R – between a trajectory and a language is trained given paired data using supervised learning.

3.1.1   Network Architecture

The inputs to the network consist of a trajectory and a natural language description. Representing the trajectory using a single sequence of frames may be prone to perceptual aliasing and occlusion. Thus, our network architecture is designed to take multiple views as inputs. We use three different viewpoints in our experiments (see Figure 5 in the Appendix), but it is straightforward to generalize to more or fewer viewpoints. In our ablation experiments, we compare the model described here with a model that takes a single viewpoint as input.
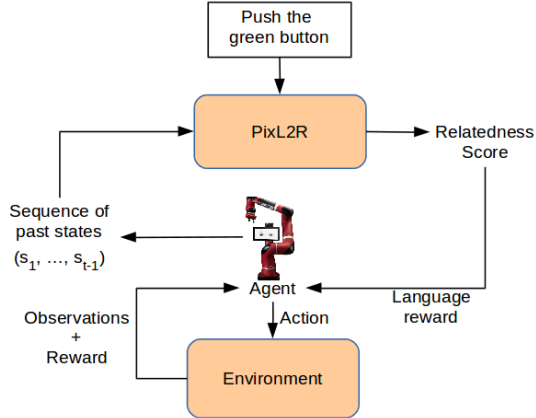
An independent CNN is used for encoding the sequence of frames from each viewpoint to generate a fixed size representation for each frame. These sequence of vectors are concatenated across the views to generate a single sequence of fixed size vectors, which is then passed through a two-layer LSTM to get an encoding of the entire trajectory.

The language description is converted to a one-hot representation, and passed through an embedding layer, followed by a two-layer LSTM. The outputs of the LSTMs encoding the trajectory and the language are then concatenated, and passed through a sequence of fully-connected layers to generate a relatedness score. See Figure 6 in the Appendix for a diagram of the neural network.

3.1.2 Data Augmentation

Frame dropping.   After sampling a trajectory, each frame is independently selected with a probability of 0.1. The resulting sequence of frames is passed through the network. This makes the training faster by reducing the input size, as well as making the network robust to minor variations in trajectories. During policy training, the trajectories are subsampled to keep 1 frame in every 10.

Partial trajectories.   Since during policy training the model will have to make predictions for partial trajectories, we use partial trajectories during supervised training as well. Given a trajectory of length L, we sample $l \sim Uniform\{1, \ldots, L\}$, and use the first l frames of the trajectory.

3.1.3 Training Objectives

Classification.   First, we trained the neural network using binary classification. The final output of the network is a two-dimensional vector, corresponding to the logits for the two classes – `RELATED` and `UNRELATED`. The network is trained to minimize the cross-entropy loss.

As mentioned above, we train the model with partial trajectories of different lengths to better match the distribution of trajectories that will be seen during policy learning. However, partial trajectories might sometimes be hard to classify as related or unrelated to the description, since it requires extrapolating the complete path the agent will follow. Our preliminary experiments suggest that these harder to classify examples affect learning – on unseen complete trajectories, a model trained with only complete trajectories has a lower error compared to a model trained on both complete and partial trajectories. This motivated us to experiment with an alternative regression setting described next.

Regression.   In this setting, the model predicts a single relatedness score between the given trajectory and language, which is mapped to $[-1, 1]$ using the $\tanh()$ function. The ground truth score is defined as $s \cdot \frac{l}{L}$, where $s = 1$ for `RELATED` and $s = -1$ for `UNRELATED` pairs, l is the length of the incomplete trajectory and L is the length of the complete trajectory as described above. Thus, given a description, a complete related trajectory has a ground truth score of 1, while a complete unrelated trajectory has a score of -1. Shorter trajectories smoothly interpolate between these values, with very small trajectories having a score close to 0. The network is trained to minimize the mean squared error. Intuitively, this results in a small loss when the model predicts the incorrect sign on short trajectories. As the trajectories become longer, incorrect sign predictions result in higher losses.

The network is trained end-to-end using an Adam optimizer [28]. We started by tuning the learning rate on a few different architectures – of the 3 values we tried (1E-3, 1E-4, 1E-5), we found 1E-4 to work the best. For the network architecture, we had 4 hyperparameters – D_1, D_2, D_3, D_4 – as shown in Figure 6. For each of these hyperparameters, we searched over the following values – $\{64, 96, 128, 192, 256, 384, 512\}$. We experimented with 8 different combinations of values for the hyperparameters using random search, and selected the model with the best performance on the validation set. The source code and the data are available at https://github.com/prasoongoyal/PixL2R.

3.2 Policy Learning Phase

Having learned a PixL2R model as described above, the relatedness scores from the model can be used to generate language-based intermediate rewards during policy learning on new scenarios. During policy training, the agent receives a natural language description of the goal, in addition to the extrinsic reward from the environment. At every timestep, the PixL2R model is used to score trajectories executed by the agent against the given natural language description, to generate intermediate rewards. We used potential-based shaping rewards [27], which are of the form $F(s_t) = \gamma \cdot \phi(s_t) - \phi(s_{t-1})$,

where s_t is the state at timestep t and $\phi : S \to \mathbb{R}$ is a potential function. In our case, s_t is the sequence of states encountered by the agent up to timestep t in the current episode. Ng et al. [27] and Grzes et al. [29] show that potential-based shaping rewards do not change the optimal policy, i.e., the optimal policies under the original reward function R and the new reward $R + F$ are identical.

For the classification setting, we used the potential function $\phi(s_t) = p_R(s_t) - p_U(s_t)$, where p_R and p_U are the probabilities assigned by the model to the classes RELATED and UNRELATED respectively. For the regression setting, the relatedness score predicted by the model is directly used as the potential for the state. Note that for both the settings, the potential of any state lies in $[-1, 1]$.

4 Domain and Dataset

4.1 Description of the Domain

We use Meta-World [8], a recently proposed benchmark for meta-reinforcement learning, which consists of a simulated Sawyer robot and everyday objects such as a faucet, windows, coffee machine, etc. Tasks in this domain involve the robot interacting with these objects, such as turning the faucet clockwise, opening the window, pressing the button on the coffee machine, etc. Completing these tasks requires learning a policy for continuous control in a 4-dimensional space (3 dimensions for the end-effector position, and the fourth dimension for the force on the gripper). While the original task suite consists of only one object in every task, we create new environments which contain one or more objects in the scene, and the robot needs to interact with a pre-selected object amongst those. In a sparse reward setting, the agent is given a non-zero reward only on successfully interacting with the pre-selected object. In the absence of any other learning signal, the agent might have to learn to approach and interact with multiple objects in the scene in order to figure out the correct object. Using natural language to describe the task in addition to the sparse reward helps alleviate this issue.

4.2 Data Collection

First, 13 tasks were selected from the Meta-World task suite. This gave us a total of 9 objects to interact with (for 4 objects, multiple tasks can be defined, e.g. turning a faucet clockwise or counter-clockwise). We then created 100 scenarios for each task as follows: In each scenario, the task-relevant object is placed at a random location on the table. Then, a new random location is sampled, and one of the remaining objects is placed at this position. This process is repeated until the new random location is close to an already placed object. This results in 1300 scenarios in total, with a variable number of objects in each scenario.

A policy was trained for each of these scenarios independently using PPO [30], which was then used to generate one video of the robot completing the task in the scenario. For this purpose, we used the dense rewards defined in the original Meta-World benchmark for various tasks. The median length of trajectories across all generated videos is 131 frames. Note that our algorithm does not need the policies used to generate the videos, so they could also be collected using human demonstrations.

To collect English descriptions of these tasks, Amazon Mechanical Turk (AMT) was used. The workers were first provided with the instructions and an example trajectory with a possible description. They were then shown a video and were given 4 possible descriptions to choose from. Only workers that passed this basic test were allowed to provide descriptions for the main tasks.[1] Each worker was asked to provide descriptions for 5 videos, which were sampled from the 1300 scenarios with the constraint that no two videos in the selected videos belong to the same task. We used simple heuristics (such as number of words and characters in the descriptions) to automatically filter out clearly bad descriptions.

Interestingly, most of the descriptions involve only the object being manipulated, with no reference to other objects in the scene. As such, a description collected for one scenario for a task can be paired with any of the 100 scenarios for the corresponding task. Therefore, we collected a total of 520 descriptions, which gives us 40 descriptions per task on average.

For each task, 79 scenarios were used for training, 18 for validation, and 3 for testing. Similarly, the descriptions for each task were split as follows – 5 for validation, 3 for testing, and the remaining for training (since there could be variable number of descriptions per task).

---

[1]The objects used for the example and the test are different from those used in the main tasks.

The distribution of number of words per description is shown in Figure 3. Our dataset contains 264 unique words, while our training set contains 248 unique words. See the Appendix (Section B) for sample descriptions and more details about the data collection process.

Given pairs of related ⟨trajectory, language⟩, positive examples were generated by pairing a scenario for one of the 13 tasks with a randomly sampled description of the corresponding task. To generate negative examples, if a scenario contains more than one object, then it was paired with the description of the task corresponding to one of the alternate objects in the scene; if there was only one object in the scene, then it was paired with the description

Figure 3: Distribution of number of words per description in the collected dataset.

of any of the remaining 12 tasks. Using such a scheme for generating negative examples is important because naively creating pairs of trajectories with descriptions of any other task randomly might result in most negative examples lacking the task-relevant object mentioned in the description. As such, the network might learn to use the *presence* of the mentioned object to compute relatedness, instead of whether the mentioned object is being *interacted with*.
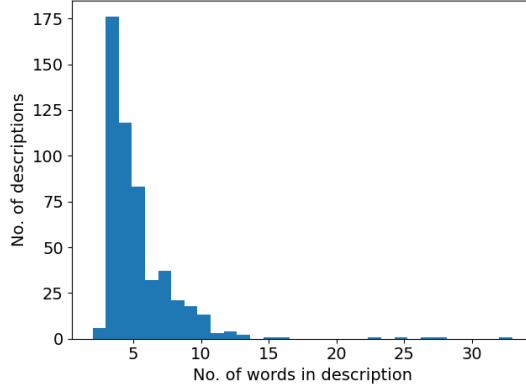
5 Experiments

5.1 Policy Training with Language-based Rewards

To empirically evaluate the effectiveness of PixL2R, the following setup was used. For each of the 13 tasks, a policy was trained for the 3 test scenarios using the PPO algorithm. Each policy training was run for 500,000 timesteps, and the number of successful completions of the task were recorded. The maximum episode length was restricted to 500 timesteps. The robot's end-effector was set to a random position within a predefined region at the beginning of each episode.

First, policy training was run with 15 random seeds, both in the sparse reward setting (`Sparse`; 1 if the agent reaches the goal, and 0 otherwise) and the hand-designed dense reward setting (`Dense`; defined in the original Meta-World benchmark). Then, a Kruskal-Wallis test was used for each scenario to identify scenarios where the number of successful episodes using dense rewards was statistically significantly more than the number of successful episodes using sparse rewards. All subsequent comparisons were done on the 16 (out of 39) scenarios for which this was true. Intuitively, these 16 tasks are too difficult to learn from sparse rewards, while they can be learned using dense rewards. Therefore, language-based dense rewards should be useful on these tasks. The remaining tasks are presumably either too simple that they can be learned with sparse rewards alone, or are too difficult to learn within 500,000 timesteps even with hand-designed dense rewards.

Then, for each of the 16 selected scenarios, a policy was trained with language-based rewards using the regression setting, in addition to the sparse rewards (`Sparse+RGR`). For each scenario, 5 policies were trained with different seeds for each of the 3 test descriptions, resulting in a total of 15 policy training runs per scenario.

A comparison of policy training curves for `Sparse` and `Sparse+RGR` rewards is shown in Figure 4 (left). Each curve is obtained by averaging over all runs (16 scenarios × 15 runs per scenario) for that reward type. The results verify that using language-based rewards in addition to sparse rewards result in higher performance on average than using only sparse ones.

Next, language-based rewards were used in addition to hand-designed rewards using a similar methodology, and the corresponding learning curves for `Dense` and `Dense+RGR` are shown in Figure 4 (right). Interestingly, we find that using language-based rewards in conjunction with hand-designed rewards result in an improvement even over hand-designed rewards. A plausible explanation is that the hand-designed dense rewards in Meta-World are suboptimal, since the reward function for each task consists of parameters that require tuning, highlighting the complexity of reward design mentioned in the introduction. This result motivates a novel paradigm wherein coarse dense rewards
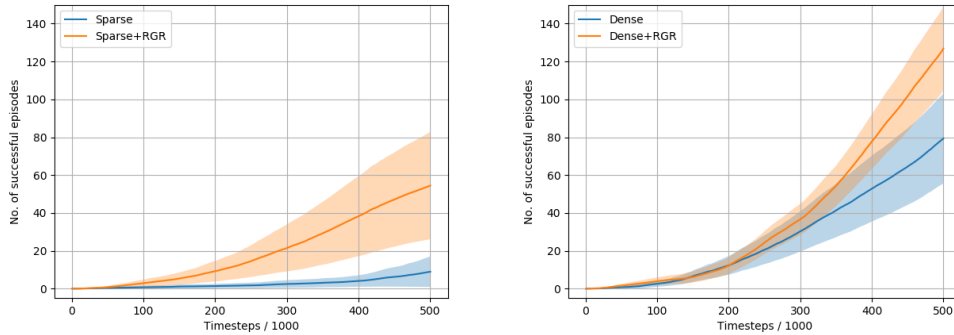
Figure 4: A comparison of policy training curves for different reward models. The shaded regions denote 95% confidence intervals.

could be designed by hand, and the proposed framework can be used to get a further improvement in policy training efficiency by using natural language.

Further, the statistical significance was computed to compare the reward functions. For each type of reward, first the average number of successful episodes was computed across all the 15 runs for each scenario, giving 16 mean successful episode scores per reward type. Since the number of successful episodes across different scenarios vary quite a bit, the mean scores for each scenario were scaled to be at most 1, by dividing by the maximum value of the mean score across all reward types for that scenario (including the reward types used in ablation experiments described in Section 5.2).

A Wilcoxon signed-rank test was then performed between the sets of normalized scores across reward types. `Sparse+RGR` was found to be statistically significantly better than `Sparse` (p-value=0.007) and `Dense+RGR` was found to be statistically significantly better than `Dense` (p-value=0.034) rewards, at a 5% significance level . Thus, the proposed approach can be used to make policy learning more sample efficient in both sparse and dense reward settings.

Having established that policy learning works better with the language-based rewards, we ran ablation experiments (described below) and analyzed the supervised learning phase (see Section C of the Appendix) to better understand our design choices and to inspect what factors most affect the efficiency of policy learning.

5.2 Ablations

All the ablation experiments were performed with language-based rewards added to dense rewards, since most applications of RL in robotics currently use dense hand-designed rewards (which could be suboptimal for complex tasks).

(1) **LastFrame**: To analyze whether using the full sequence of frames contains more information than the last frame, instead of using the sequence of frames in the trajectory, only the last frame of the trajectory was used, both for training the PixL2R model, as well as for policy training.

(2) **MeanpoolLang**: To study if the temporal ordering of the words in the description is useful, the LSTM used to encode the language was replaced with the mean-pooling operation.

(3) **MeanpoolTraj**: To study if the temporal ordering of the frames in the trajectory was useful, the LSTM used to encode the sequence of frames was replaced with the mean-pooling operation.

Setting	Mean Successful Episodes	p-value w.r.t. Dense
Dense	79.4	-
Dense+RGR	126.9	0.0340
LastFrame	133.5	0.0114
MeanpoolLang	138.3	0.0004
MeanpoolTraj	78.4	0.9601
SingleView	100.4	0.3789
Dense+CLS	102.0	0.6384

Table 1: Comparison of various ablations to the Dense+RGR model.

(4) **SingleView**: To study the impact of perceptual aliasing and/or occlusion when using a single viewpoint, instead of using 3 viewpoints for the trajectory, only 1 viewpoint was used. A model was trained with *each* of the three viewpoints in the supervised learning phase, and the model with the

best validation score was used for policy learning.

(5) **Dense+CLS**: Instead of the regression loss, classification loss was used, to understand the benefit of using regression loss when working with partial trajectories.

For each ablation, the same setup was used as for `Dense+RGR`. This model is used to generate rewards for policy training, for each of the 16 scenarios with 5 random seeds for all the 3 descriptions as before. The mean successful episodes across all runs are reported in Table 1. Further, the p-values for Wilcoxon tests between each ablation and the `Dense` rewards is reported, from which we can make the following observations:

- Using only the last frame (`LastFrame`), or using mean-pooling instead of an LSTM to encode the language (`MeanpoolLang`) does not substantially affect policy learning efficiency. In both these cases, the resulting model is still statistically significantly better than `Dense` rewards. Both of these results agree with intuition, since the progress in the task can be predicted using the last frame alone, and since the linguistic descriptions are not particularly complex in the given domain, simply looking at which words are present or absent is often sufficient to identify the task without using the ordering information between the words.
- Using mean-pooling instead of an LSTM to encode the sequence of frames (`MeanpoolTraj`) drastically reduces the number of successful episodes, and results in no statistically significant improvement over `Dense`. Again, this agrees with intuition, since it is not possible to infer the direction of movement of the robot from an unordered set of frames.
- Using a single view instead of multiple views (`SingleView`) results in a smaller increase in the number of successful episodes, which is no longer statistically significant over `Dense`. As mentioned earlier, using frames to represent trajectories requires addressing challenges such as perceptual aliasing and occlusion, and these ablation results suggest that using multiple viewpoints alleviates these issues.
- Using classification loss instead of regression (`Dense+CLS`) also leads to a drop in performance, again making the resulting improvements no longer statistically significant. This is consistent with our initial observation (Section 3.1.3), wherein, the learning problem becomes more difficult due to partial trajectories when the classification loss is used.

It is worth noting that while these ablations agree with intuition, and therefore suggest that the model is extracting meaningful information from trajectories and language descriptions, the performance of these variants depends crucially on the domain. For instance, an environment that is not fully observable in the last frame might show a significant drop in performance when using only the last frame instead of the full trajectory.

6   Conclusion

We proposed an approach for mapping pixels to rewards, conditioned on a free-form natural language description of the task. Given paired ⟨trajectory, language⟩ data, first, a relatedness model – PixL2R – is learned between a sequence of states and a natural language description using supervised learning. This model is then used to generate intermediate rewards for policy learning using a natural language task description. Our experiments on a simulated robot manipulation domain show that the proposed approach can significantly speed up policy learning, both in sparse and dense reward settings. The proposed technique can be used in a novel RL training paradigm, wherein language-based rewards can be used to make training efficient over coarse hand-designed dense rewards.

The proposed approach can be extended in multiple ways. First, further experimentation on a richer domain could be used to analyze generalization to new tasks with novel compositions of objects and actions seen during training. Secondly, the current model only works for a single instruction and could be extended to use a sequence of instructions, for instance, by starting with the first instruction in the sequence, and transitioning to the next instruction when the prediction of the PixL2R model is above a threshold. Next, PixL2R currently encodes the trajectory and language independently, which are then concatenated to obtain a relatedness score. For more complex domains, it might be helpful to use an attention-based model to learn a mapping between spatio-temporal regions of the trajectory and words or phrases in the language. Finally, it may be useful to fine-tune the PixL2R model on trajectories seen during policy learning.

Acknowledgements

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] Y. Gao, J. Peters, A. Tsourdos, S. Zhifei, and E. M. Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 2012.

[3] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktaschel. A survey of reinforcement learning informed by natural language. In *IJCAI 2019: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, August 2019.

[4] J. MacGlashan, M. Littman, R. Loftin, B. Peng, D. Roberts, and M. E. Taylor. Training an agent to ground commands with reward and punishment. In *Proceedings of the AAAI Machine Learning for Interactive Systems Workshop*, 2014.

[5] E. C. Williams, N. Gopalan, M. Rhee, and S. Tellex. Learning to parse natural language to grounded reward functions with weak supervision. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.

[6] D. Bahdanau, F. Hill, J. Leike, E. Hughes, A. Hosseini, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

[7] P. Goyal, S. Niekum, and R. J. Mooney. Using natural language for reward shaping in reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, August 2019.

[8] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[9] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

[10] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, volume 1, page 2, 2011.

[11] J. Sung, S. H. Jin, and A. Saxena. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. In *Robotics Research*, pages 701–720. Springer, 2018.

[12] D. Arumugam, S. Karamcheti, N. Gopalan, L. L. Wong, and S. Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. *Robotics: Science and Systems (RSS)*, 2017.

[13] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*. San Jose, CA, 2004.

[14] S. Branavan, D. Silver, and R. Barzilay. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

[15] R. Kaplan, C. Sauer, and A. Sosa. Beating atari with natural language guided reinforcement learning. *arXiv preprint arXiv:1704.05539*, 2017.

[16] N. Waytowich, S. L. Barton, V. Lawhern, E. Stump, and G. Warnell. Grounding natural language commands to starcraft ii game states for narration-guided reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 110060S. International Society for Optics and Photonics, 2019.

[17] S. Branavan, N. Kushman, T. Lei, and R. Barzilay. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 126–135. Association for Computational Linguistics, 2012.

[18] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018.

[19] J. Fu, A. Korattikara, S. Levine, and S. Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *International Conference on Learning Representations (ICLR)*, 2019.

[20] K. Narasimhan, T. Kulkarni, and R. Barzilay. Language understanding for text-based games using deep reinforcement learning. *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[21] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *Conference on Robot Learning (CoRL)*, 2019.

[22] B. Harrison, U. Ehsan, and M. O. Riedl. Guiding reinforcement learning exploration using natural language. *arXiv preprint arXiv:1707.08616*, 2017.

[23] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[24] J. Andreas, D. Klein, and S. Levine. Learning with latent language. *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[25] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox. Prospection: Interpretable plans from language by predicting the future. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6942–6948. IEEE, 2019.

[26] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[27] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, volume 99, pages 278–287, 1999.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations (ICLR)*, 2015.

[29] M. Grzes. Reward shaping in episodic reinforcement learning. 2017.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *arXiv preprint arXiv:1907.03146*, 2019.

A   Approach Details

Figure 5 shows the viewpoints used in our experiments. Figure 6 shows a diagram of the neural network architecture described in Section 3.1.1.
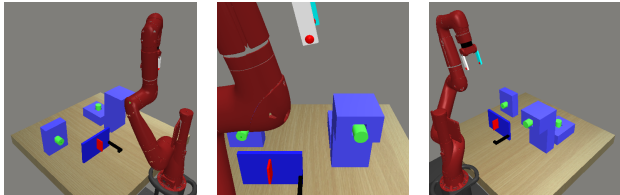

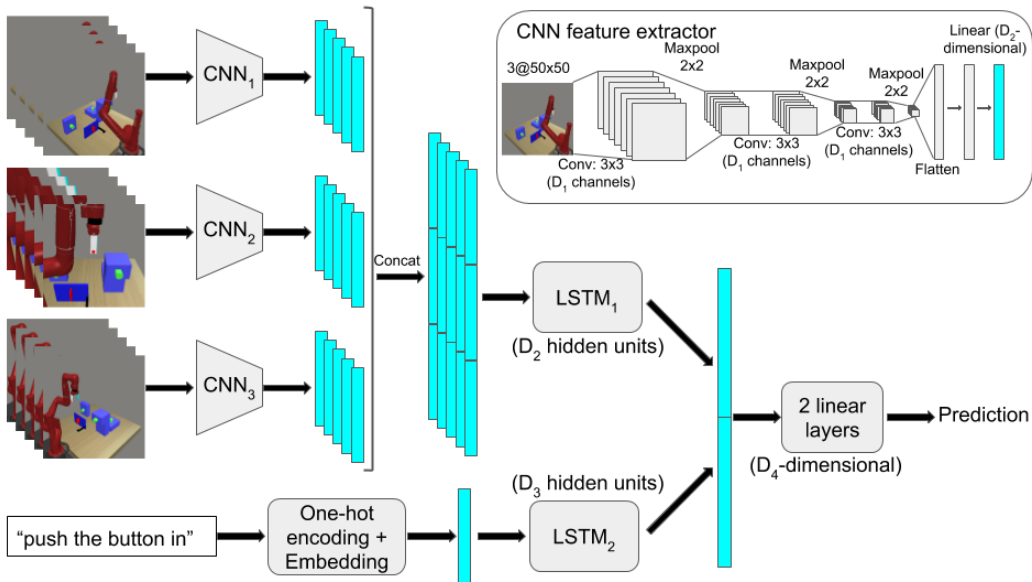Figure 5: Viewpoints used for data collection and experiments.

Figure 6: Neural network architecture: The sequence of frames from the three viewpoints are passed through three separate CNN feature extractors. The resulting feature vectors are concatenated across views. The sequence is then passed through an LSTM to obtain an encoding of the trajectory. The given linguistic description is converted to one-hot representation, and passed through an embedding layer, followed by an LSTM. The outputs of the two LSTMs is concatenated and passed through a sequence of 2 linear layers (with a ReLU activation between them) to generate the final prediction.

B   Data Collection Details

Since the models of the objects in the environment are coarse, it is usually non-trivial to recognize the real-world objects they represent from the models alone. To guide the AMT workers to use the names of real-world objects the models represent, we showed a table of the models with prototypical images of real-world objects that closely match the models (shown in Figure 7). This enabled us to get descriptions that use the real-world object names, without priming the workers with specific words.[2]

Some examples of descriptions collected are shown in Table 2.

---

[2]Despite using this technique, we still got some responses where people described the models directly instead of using the object names, e.g. "Pull the red box out slightly in blue square." instead of using the word *toaster*.
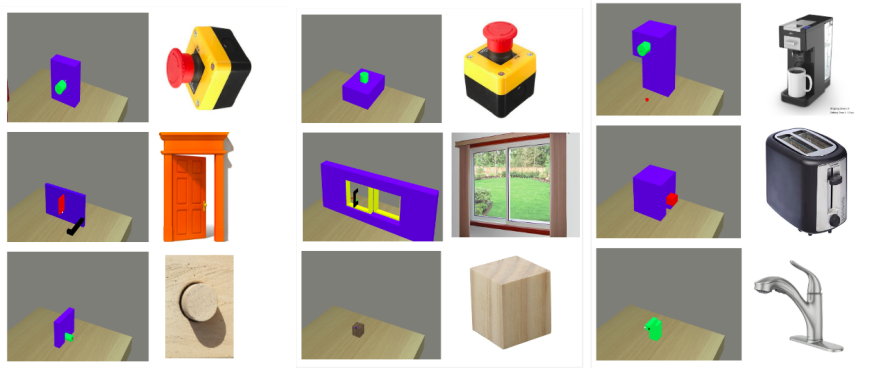
Figure 7: List of objects used

Task Id	Description
0	Press the button.
0	Pressing the button
1	Push peg in to hole.
1	Push the green button.
2	Turn on the coffee maker
2	push in the green button
3	Push toaster handle down
3	Push down the red block.
4	pressing down the object
4	pull down the red switch
5	move the plate down
5	push down the slider
6	Close the door
6	Open the door.
7	twisting the cube
7	rotate the object
8	Rotate the lever anticlockwise
8	Turn the faucet to the right.
9	rotating the object
9	turn on the faucet
10	Open the window.
10	Open the yellow window.
11	Slide the window to the left.
11	Close the Window.
12	pull out the green block
12	Pull out the green piece

Table 2: Examples of descriptions collected using AMT.

C   Additional Experiments: Word-level Analysis

In order to understand how the supervised learning phase is using different words in the description, the supervised model was used to make predictions on the test set, and the gradient of the loss was computed with respect to the continuous representation of the words in the descriptions (i.e. after the embedding layer). The mean of the absolute values of these gradients is then a measure of how much the prediction is affected by the corresponding word. The values are reported in Table 3, which were scaled so that the maximum value for any description is 1.

First, we observe that for all the descriptions, the words describing the main object have a very high average gradient magnitude – *green* and *button* in description 1, *red* and *block* in description 2, *lever* and *toaster* in description 3, *faucet* in description 4, *green* and *lever* in description 5, and *window* in description 6. Several verbs also have a high average gradient magnitude – *turn on* in description

	Descriptions						
	Average magnitude of gradient for each word						
1.	push	the	green	button			
	0.53	0.30	1.00	0.94			
2.	push	down	the	red	block		
	0.42	0.57	0.34	1.00	0.91		
3.	pull	down	the	lever	on	the	toaster
	0.16	0.31	0.15	0.75	0.58	0.36	1.00
4.	turn	on	the	faucet			
	0.94	1.00	0.44	0.87			
5.	slide	the	green	lever	to	the	left
	0.52	0.23	0.94	1.00	0.77	0.30	0.78
6.	open	the	window				
	0.83	0.32	1.00				

Table 3: Average magnitude of gradients for different words in a description for the relatedness score prediction.

4 and *open* in window. Verbs in other descriptions do not have a high gradient magnitude because for those descriptions, the object affords only one possible interaction, thus making the verb less discriminatory. For the objects *faucet* and *window*, there are two possible actions each (*turning the faucet on or off* and *opening or closing the window*); thus the verb also carries useful information for these objects.

This analysis suggests that the model learns to identify the most salient words in the description that are useful to predict the relatedness between a trajectory and language.